

DİZİLER / ARRAY

▼ Dizi nedir?

- Tek bir değişken altında birden fazla aynı türde değeri toplamamızı sağlayan veri kümelerine dizi denmektedir.
- Prosedürel programların veri kümesidir. Yani yazılımsal boyutta yazılım adına RAM'de birden fazla değeri tek bir değişken altında bir veri kümesi halinde tutabilirler.
- DİZİLER REFERANS TÜRLÜ DEĞERLERDİR. YANI NESNESEL YAPILARDIR. ÖZLERİNDE CLASSTIRLAR.
- Yapısal olarak Ram'de HEAP'te tutulur.
 - A'nı yaşını tutuyosam stack de tutuyoruz. A'nın B'nin C'nin yaşını tutuyosak Heap'te
- Prototipi → `type [] name = new type [.....]`

Diziler/Arrays

Diziler içerisinde birden fazla aynı türde değer tutabilen yapılardır. Prosedürel programlamanın temel veri kümeleridir. Yani yazılımsal boyutta, yazılım adına RAM'de birden fazla değeri tek bir değişken altında bir veri kümesi halinde tutabilirler...

Diziler, veri kümeleri oldukları için, içerindeki birden fazla veri üzerinde kümesel işlemler yapmamızı sağlayabilirler...

Diziler, prosedürel prog. temel yapıları oldukları için daha gelişmiş yapılar olan koleksiyonlarında temsiliyetini temsil etmektedirler ve gelişmelerine katkıda bulunmaktadır.

Diziler referans türü değerlerdir. Yani nesnel yapılardır. Özlerinde classtırlar...

Yapısal olarak RAM'de Heap'te tutulurlar...

KRİTİK

Diziler içerisinde eleman/değer eklendikçe bunları sırasıyla bir şekilde depolamaz. Düzenli bir şekilde sıralı depolayacaktır.

Dizilerde eklenen elemanlar **index** ismini verdiğimiz numaralarla ardışık bir şekilde numaralandırılırlar...

index no : Her bir elemana verilen ardışık sayı. 0'dan başlar n-1'e kadar gider.

Dizi içerisinde her türlü değer koyulabilir. Değer türü-Referans türü

Bir dizi sade ve sadece tek bir türde değer alabilir.

Dizi içerisinde kullanılan değerler işlevsel olarak aynı mahiyette olmalıdır. Örneğin; yaş dizisine maaş bilgisi aynı türde olduğu halde verilmemelidir.

PROTİP

type a: ---> Değişken

type [] b ---> Dizi

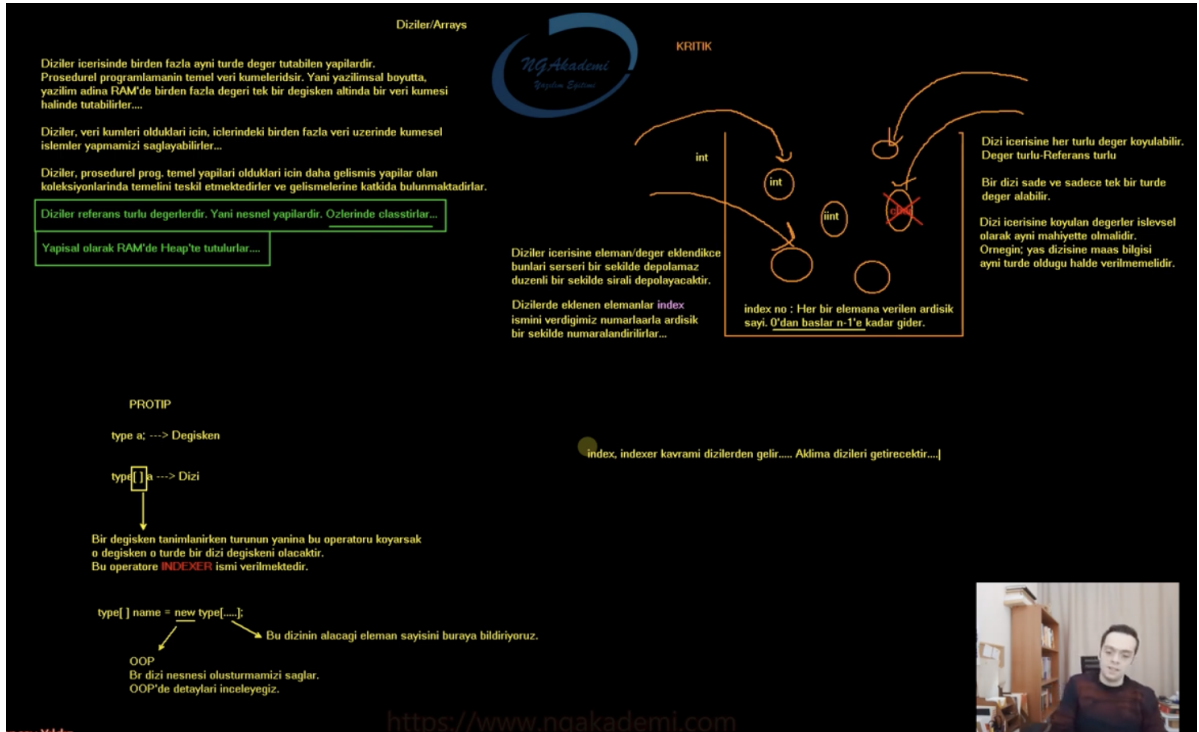
Bir değişken tanımlanırken türünün yanına bu operatörü koyarsak o değişken o türde bir dizi değişkeni olacaktır. Bu operatöre **INDEXER** ismi verilmektedir.

type [] name = new type [.....];

Bu dizinin alacağı eleman sayısını buraya bildiriyoruz.

OOP
Bir dizi nesnesi oluşturmanızı sağlar.
OOP'de detayları inceleyeceğiz.

<https://www.nca akademi.com>



▼ Dizi nasıl tanımlanır?

```
{
    #region Sınırlılıklar
    //Dizilerde tanımlşama yaparken eleman sayısının bildirilmesi zorunluluđu bir sınırlılıktır.
    //
    #endregion
    #region Diziler - Array
    //Tek bir deđişken altında birden fazla "aynı türde" deđeri toplamamızı sađlayan veri kümelerine dizi
    denmektedir.
    int[] yaslar = new int[7];
    #endregion
}
```

Bir dizi tanımlaması yapıldığı an bellekte o diziye kullansakta kullanmasakta verilen eleman sayısı kadar alan tahsisinde bulunulur....

Dizilerde tanımlama yapıldığı an alan tahsisinde bulunulması bizim için pekte doğru bir durum değildir.

Kullanılmadığı halde diziler direkt olarak bellekten belirtilen eleman sayısı kadar alan tahsisinde bulunması bir sınırlılıktır.

— Diziler alan tahsisi yapıldığında ilgili alanlara turune uygun default deđerleri atarlar.

STACK

yaslar
0
0
0
0
0
0
0

HEAP

obj(int[] array)

```
#region Sınırlılıklar
//Dizilerde tanımlşama yaparken eleman sayısının bildirilmesi zorunluluđu bir sınırlılıktır.
//Diziler tanımlandığında kullansakta kullanmasakta bellekte belirtilen eleman sayısı kadar alan tahsisinde
    bulunurlar. Bu durumda bellek boyunda ekstradan maliyete sebep olacađı için bir sınırlılıktır.
#endregion
#region Diziler - Array
//Tek bir deđişken altında birden fazla "aynı türde" deđeri toplamamızı sađlayan veri kümelerine dizi
    denmektedir.
|
```

▼ Tanımlanan diziye nasıl deđer atanır?

birden fazla "aynı türde" değeri

Index numarası dizilere sistem tarafından otomatik verilen ve kimlik mahiyetinde kullanılabileceğimiz bir numaradır.

Her bir elemana karşılık gelen unique değerlerdir. Haliyle böyle olması demek her bir elemana istediğimiz zaman erişip, değer atama yapabilmemizi yahut var olan değeri okuyabilmemizi sağlamaktadır.

Dolayısıyla dizilerde tanımlanmış alanlara/elemanlara değer atama yahut değer okuma operasyonları index numaraları eşliğinde gerçekleştirilir.

index	value
0	0
1	0
2	0
3	0
4	0
5	0
6	0

- Tanımlanmış diziden değer okuma
 - Değer aralığının aşılmasına dikkat etmeliyiz.
 -
- ▼ Dizi içerisinde döngüyle dönmek

```

string[] personeller = new string[10];
personeller[0] = "Hilmi";
personeller[1] = "Hüseyin";
personeller[2] = "Rıfkı";
personeller[3] = "Şuayip";
personeller[4] = "Muiddin";
personeller[5] = "Naci";
personeller[6] = "Hüsnü";
personeller[7] = "Nurullah";
personeller[8] = "Cabbar";
personeller[9] = "Akif";

//for (int i = 0; i < 10; i++)
//{
//    Console.WriteLine(personeller[i]);
//}

int i = 0;
do
{
    Console.WriteLine(personeller[i++]);
} while (i < 10);

//Console.WriteLine(personeller[0]);
//Console.WriteLine(personeller[1]);
//Console.WriteLine(personeller[2]);
//Console.WriteLine(personeller[3]);
//Console.WriteLine(personeller[4]);
//Console.WriteLine(personeller[5]);

```

- Diziden gelen eleman sayısı ile şart belirlenir, eleman sayısından fazla bir şart koymamız, algoritmanın patlamasına sebebiyet verir.
 - Bu durumu yaşamamak için eleman sayısını bilemediğimiz durumlarda, .Length özelliğini kullanabiliriz. Örneğin;

```
for (int i=0 ; i <personeller.Length ; i++)
```

şeklinde bir kullanım, manuel bir bağımlılığı ortadan kaldırır.

▼ Sınırlılıklar

```
#region Sınırlılıklar
//Dizilerde tanımlşama yaparken eleman sayısının bildirilmesi zorunluluğu bir
    sınırlılıktır.
//Diziler tanımlandığında kullansakta kullanmasakta bellekte belirtilen eleman sayısı
    kadar alan tahsisinde bulunurlar. Bu durumda bellek boyunda ekstradan maliyete
    sebep olacağı için bir sınırlılıktır.
//Dizilerde eleman sayısının başta belirlenmesi durumunda, ihtiyaca binaen daha fazla
    değer atamak istediğimizde bu değerleri atayamacağımızdan ve dizinin aralığını
    genişletemeyeceğimizden dolayı bu durum bir sınırlılık olarak karşımıza
    çıkmaktadır.
//Dizilerde elemanlara değer atarken indexer operatörüyle çok haşır neşir
    olunmaktadır. Bu durumda bir sınırlılıktır.
#endregion
```

▼ Dizi tanımlama varyasyonları

```
#region 1. Varyasyon
int[] yaslar = new int[3];
yaslar[2] = 123;
Console.WriteLine(yaslar[2]);
#endregion
#region 2. Varyasyon
int[] yaslar2 = { 30, 25, 41, 52 };
string[] isimler = { "Rıfkı", "Şuayip", "Hüseyin", "Hilmi", "Mehmet" };
#endregion
```

```
#region 3. Varyasyon
string[] isimler2 = new string[] { "Rıfkı", "Şuayip", "Hüseyin", "Hilmi", "Mehmet", "klqmsdkamskdm" };
#endregion
#region 4. Varyasyon
string[] isimler3 = new string[3] { "pkasmdkmasd", "ksmdfkmsdfk", "kmkasmdkdm" };
#endregion
```

```
#region 5. Varyasyon
//int[] sayilar = new[] { 3, 5, 7 };
//var sayilar2 = new[] { "3", "4", "2", "7", "lkasmdkmasd" };
#endregion
```

▼ Array sınıfı

▼ Nedir?

- Dizi= array sınıfı
- Dizi olarak tanımlanan değişkenler Array sınıfından türetilmektedir.

```
int [] yaslar= new int[3]; //ile
//Bu şekilde çalışıldığında ilgili diziye verisel müdahale
//Genellikle bu format algoritmalarda tercih edilir. Çünkü

Array yaslar= new int[3]; // aynı şey
// Bu şekilde ise fonksiyonel çözümler gerektirilmektedir
// Genellikle elimizdeki dizinin üzerinde işlem yaparken

//İkisi arasındaki fark; Dizi eğer kendi türünde referans
```

- Dizilerde Array sınıfından gelen belirli metotlar ve özellikler mevcuttur.
- Referans türlü değişkenlerin özelliği Stack'te değişkeni tutulurken bu değişkenin Heap'teki değeri/nesneyi referans etmesidir.
- Key Value tarzında çalışan koleksiyonlar **dictionary** dediğimiz ya da belirli generic yapılanmalara sahip koleksiyonlar türetilmiştir. Ya da FIFO(First In First Out), LIFO(Last In First Out) tarzında çalışan işte davranışsal olarak ilk giren son çıkar ilk giren ilk çıkar tarzında çalışan koleksiyonlar geliştirilmiştir. Ama hepsinin temelinde bu sınırlılıklardan arındırılmış gene birden fazla veriyi tutabilen yapılar vardır.
-

Bu tür dizileri genellikle operasyonel kullanırız. Yani sen bir algoritma ya bu diziyi göndereceksen o algorithmada indexer falan ihtiyacın olabilir

Array türünde çalışıyorsak genellikle ilgili dizi üstünde işlem yaparız. Diziyi sıralarız dizinin boyutunu ölçeriz. Okunabilir midir değil midir bununla ilgili bilgi alırız vs. Diziyi ilgili işlem yapıyorsan sadece dizinin üstünde diziye dair

direkt kendi referansında
gönderebilirsin.

bilgisel bir çalışma yapıyorsan Array
formatında çalışırız.

▼ Array türünden bir diziye nasıl değer atanır ve okunur?

- İlk yöntem

```
#region 1. Yöntem
int[] dizi = new int[3];
dizi[0] = 30;
dizi[1] = 31;
dizi[2] = 32;
Array dizi2 = dizi;
#endregion
```

- İkinci yöntem

```
#region 2. Yöntem
//Array dizi = new int[] { 3, 5, 7, 9 };
//Array dizi = new int[4] { 3, 5, 7, 9 };
Array dizi = new[] { 3, 5, 7, 9 };
//Array dizi = { 3, 5, 7, 9 }; //KULLANILAMAZ
#endregion
```

- Üçüncü yöntem

```
#region 3. Yöntem
Array dizi = new int[3];
dizi.SetValue(30, 0);
dizi.SetValue(31, 1);
dizi.SetValue(32, 2);
#endregion
```

- Nasıl okunur?


```
#region 3. Yöntem
Array dizi = new int[3];
dizi.SetValue(30, 0);
dizi.SetValue(31, 1);
dizi.SetValue(32, 2);

object value = dizi.GetValue(1);
Console.WriteLine(value);
#endregion
#endregion
```

Metotlar
Özellikler

▼ Metotlar

▼ Clear

- Dizi içerisindeki tüm elemanlara, dizinin türüne uygun **default değerleri atayan** bir fonksiyondur.

```
Array isimler = new[] { "Hilmi", "Hüseyin", "Şuayp" };
#region Metotlar
#region Clear
//Dizi içerisindeki tüm elemanları siliyor diye b...
elemanlara, dizinin türüne uygun default değ...
for (int i = 0; i < isimler.Length; i++)
    Console.WriteLine(isimler.GetValue(i));
Array.Clear(isimler, 0, isimler.Length);
Console.WriteLine("*****");
for (int i = 0; i < isimler.Length; i++)
    Console.WriteLine(isimler.GetValue(i));
#endregion
Copy
```

•

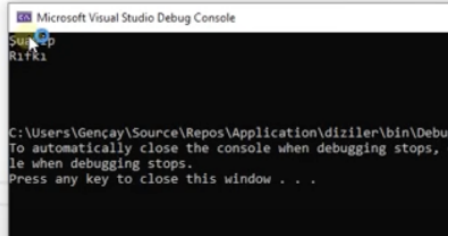
▼ Copy

- Elimizdeki bir dizinin verilerini bir başka diziye kopyalamamızı sağlayan bir fonksiyondur.

```
Array isimler = new[] { "Hilmi", "Hüseyin", "Şuayip", "Rıfkı", "Hamdullah" };
#region Metotlar
Clear
#region Copy
//Elimizdeki bir dizinin verilerini bir başka diziye kopyalamamızı sağlayan bir fonksiyondur.
string[] isimler2 = new string[isimler.Length];

//Array.Copy(isimler, isimler2, 5);
//for (int i = 0; i < isimler2.Length; i++)
//    Console.WriteLine(isimler2[i]);

Array.Copy(isimler, 2, isimler2, 0, 2);
for (int i = 0; i < isimler2.Length; i++)
    Console.WriteLine(isimler2[i]);
#endregion
```



▼ IndexOf

- Dizi içerisinde bir elemanın olup olmadığını sorgulayabildiğimiz fonksiyondur.
- Arama neticesinde ilgili değer varsa int olarak o değer index numarasını döndürecek.
- Yoksa -1 değerini döndürür.

```
Array isimler = new[] { "Hilmi", "Hüseyin", "Şuayip", "Rıfkı", "Hamdullah" };
#region Metotlar
Clear
Copy
#region IndexOf
//Dizi içerisinde bir elemanın var olup olmadığını sorgulayabildiğimiz fonksiyondur.
//Arama neticesinde ilgili değer varsa int olarak o değer index numarasını döndürür.
//yoksa -1 değerini döndürür.
//int index = Array.IndexOf(isimler, "Ali");
//if (index != -1)
//{
//    //Demek ki aranan değer ilgili dizide bulunmaktadır...
//    Console.WriteLine("Var");
//}

int index = Array.IndexOf(isimler, "Rıfkı", 1, 3);
#endregion
```

▼ Reverse

- Elimizdeki dizinin elemanlarını tersine sıralayan bir fonksiyondur.

```

IndexOf
#region Reverse
//Elimizdeki dizinin elemanlarını tersine sıralayan bir fonksiyondur.
for (int i = 0; i < isimler.Length; i++)
    Console.WriteLine(isimler.GetValue(i));

//Array.Reverse(isimler);
//Console.WriteLine("*****");

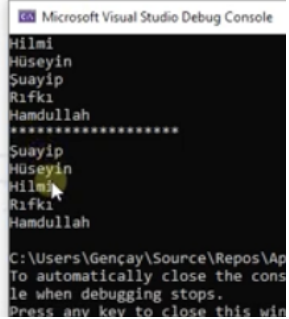
//for (int i = 0; i < isimler.Length; i++)
//    Console.WriteLine(isimler.GetValue(i));

Array.Reverse(isimler, 0, 3);
Console.WriteLine("*****");

for (int i = 0; i < isimler.Length; i++)
    Console.WriteLine(isimler.GetValue(i));

#endregion

```



•

▼ Sort

- Elimizdeki fonksiyonun sıralanmasını sağlar. Küçükten büyüğe doru.

```

Reverse
#region Sort
for (int i = 0; i < isimler.Length; i++)
    Console.WriteLine(isimler.GetValue(i));

Array.Sort(isimler);
Console.WriteLine("*****");

for (int i = 0; i < isimler.Length; i++)
    Console.WriteLine(isimler.GetValue(i));

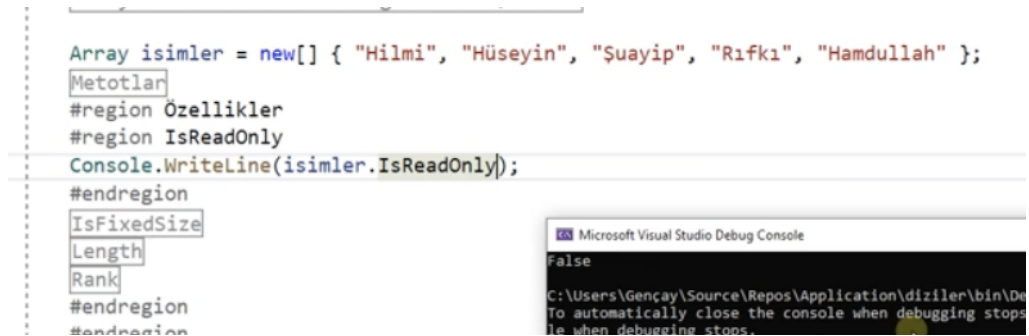
```

•

▼ Property (Özellikler)

▼ IsReadOnly

- Bir dizinin okunabilir olup olmadığını verir. True ya da false olarak.



▼ IsFixedSize

- Bir veri kümesinin eleman sayısının sabit olup olmama durumunu öğrenebiliriz.
- Tüm dizilerde eleman sayısı sabit olduğundan, sürekli true dönecektir. Örneğin arrayList koleksiyonunda false dönecektir.
-

```
Console.WriteLine(isimler.IsFixedSize))
```

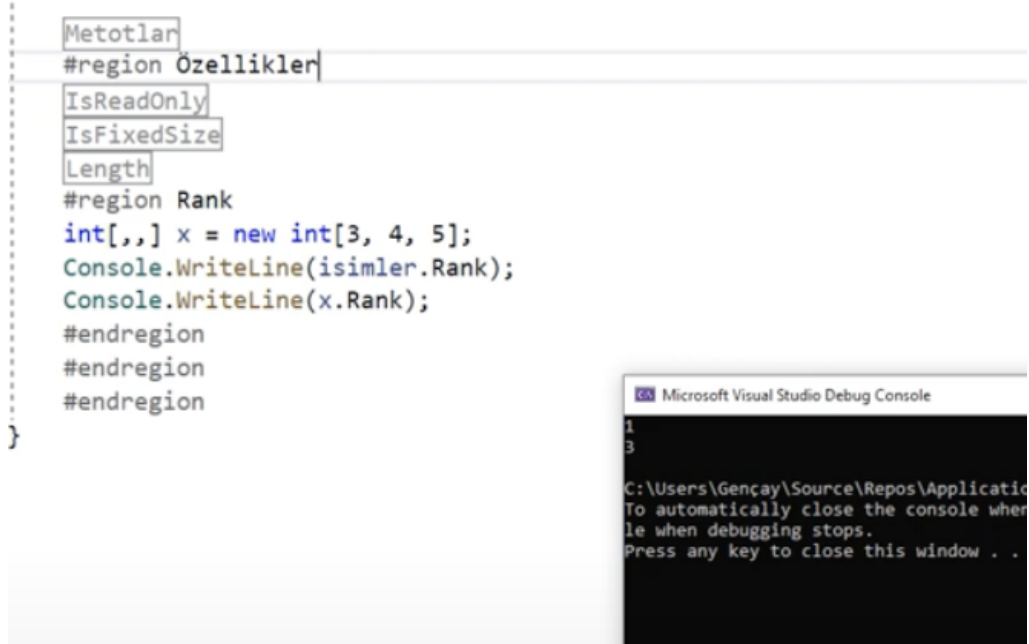
▼ Length

- Dizinin eleman sayısını verir.

```
for (int i=0 ; i <personeller.Length ; i++)
```

▼ Rank

- İlgili dizinin derece sayısını gösterir.

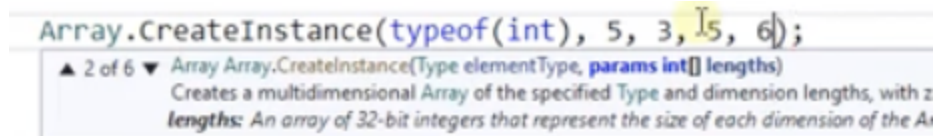


▼ CreateInstance metodu ile dizi tanımlama

```
0 references
static void Main(string[] args)
{
    int[] yaslar = new int[3];
    //Normalde yukarıdaki gibi yapılan dizi tanımlaması esasında arkaplanda Array sınıfının CreateInstance metodunu kullanmaktadır. Bizlerde bu metodu
    kullanarak fonksiyonel diziler oluşturabilmekteyiz.
    Array yaslar2 = Array.CreateInstance(typeof(int), 3);
}
```

• Çok boyutlu dizi oluşturmak

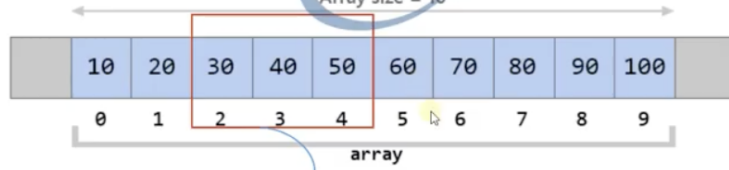
- 4 dereceli



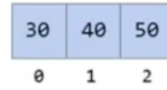
-

▼ Dizilerde verisel performans

```
int[] sayilar = { 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 };  
int[] sayilar2 = sayilar[2..5];
```



Böyle bir durumda aklımıza şu soru gelmektedir?



İlgili alan ayrı bir dizi olarak elde edilecek ve değerler RAM'de tekrardan tahsis edilecektir.

Zaten var olan dizinin dışında neden başka dizi tanımlandı? Bu ekstradan maliyet değil mi?

El cevap : Bal gibide maliyettir.

- Bunun çözümü şurada anlattık